Optical Character Recognition
Project OCR

Sommaire

1. Présentation des membres 3
2. Présentation du projet 4
   2.1. Spécification du sujet 4
   2.2. Découpage du travail 5
      2.2.1. 1ère partie : Prétraitement 5
      2.2.2. 2ème partie : Extraction 5
      2.2.3. 3ème partie : Reconnaissance des caractères 5
      2.2.4. 4ème partie : Interface graphique 6
3. Site Internet 7
4. Chronologie 10
   4.1. Chronologie (Groupe) 10
   4.2. Chronologie (individuelle) 11
      4.2.1. Xavier 11
      4.2.2. Kévin 12
      4.2.3. Jocelyn 12
      4.2.4. Loïc 13
5. Amélioration à venir 14
   5.1. Interface 14
   5.2. Réduction du bruit 14
   5.3. Détection de l'angle de rotation 14
   5.4. Rotation 14

1. Présentation des membres

Le groupe de développement est composé de 4 membres :

- Xavier Michaud : Chef de Projet

- Loïc Billard

- Jocelyn Lagarenne

- Kevin Fonlupt

Les 4 membres sont en API et découvrent le Caml cette année.

2. Présentation du projet

2.1. Spécification du sujet

Le projet que nous sommes en train de réaliser, est un logiciel de reconnaissance de caractères : un OCR (Optical Character Recognition)

Il va permettre de récupérer dans un logiciel de traitement de texte, les textes issus des documents imprimés (courriers, pages de livre, articles de presse).

Après numérisation, le logiciel attribue à chaque symbole du texte le caractère correspondant à partir d'un profil entraîné avec des exemples. Mais la transformation en texte brut d'une page numérisée n'est pas la seule fonctionnalité du logiciel. Il analyse également le mot reformé avec un dictionnaire pour en diminuer le nombre d'erreurs.

Pour cela le projet va disposer d'une interface graphique permettant de charger des images et de les traiter automatiquement.

2.2. Découpage du travail

Le projet est découpé en 4 parties majeures. La 1ère et la dernière partie sont réalisées en C, l'extraction des lignes et caractères ainsi que la reconnaissance des caractères sont quant à elles, réalisées en Ocaml.

2.2.1. 1ère partie : Prétraitement

Prétraitement de l'image de départ, numérisée à partir d'un document A4 : le but est de rendre le document « parfait », c'est-à-dire de retirer le bruit généré lors de la numérisation ainsi que d'augmenter les contrastes des caractères. Il faut aussi redresser l'image si besoin.

2.2.2. 2ème partie : Extraction

Extraction des lignes, des caractères, et détection de la mise en forme du document (nous nous contenterons, dans un premier temps de la typographie des caractères (gras, italique, souligné, etc.).

2.2.3. 3ème partie : Reconnaissance des caractères

Le but de cette partie est de reconnaître les différents caractères au travers d'un réseau de neurone.

Il faudra trouver une bonne représentation des données caractérisant l'image d'un symbole, et standardiser ces données d'entrée pour le réseau de neurones.

Il faudra également gérer le cas des caractères trop ambigus ou trop difficiles à reconnaître. Le "classifieur" devra les rejeter pour pouvoir ensuite les traiter au niveau de l'interface.

2.2.4. 4ème partie : Interface graphique

Cette interface sera relativement simple :

Une zone de texte afin de permettre à l'utilisateur de modifier le texte du document numérisé prendra la moitié de la fenêtre. (Pour le moment l'interface permet de voir l'image d'origine et l'image « parfaite » générée).

L'autre moitié du logiciel étant consacrée à la visualisation du document original afin de pouvoir vérifier la similarité des textes.

3. Site Internet

Le site est développé en php avec une base de données mysql. Il est pour le moment hébergé chez free et ne dispose pas encore pour le moment de DNS, mais nous en aurons bientôt un !

Il permet aux visiteurs de se renseigner sur notre projet et son avancement mais aussi sur les membres de l'équipe de développement.

Les visiteurs pourront aussi une fois le projet finalisé télécharger les fichiers d'installation directement sur le site. Ils auront accès à une Faq et un mode d'emploi online. Ils pourront aussi nous contacter s'ils ont des questions à nous poser.

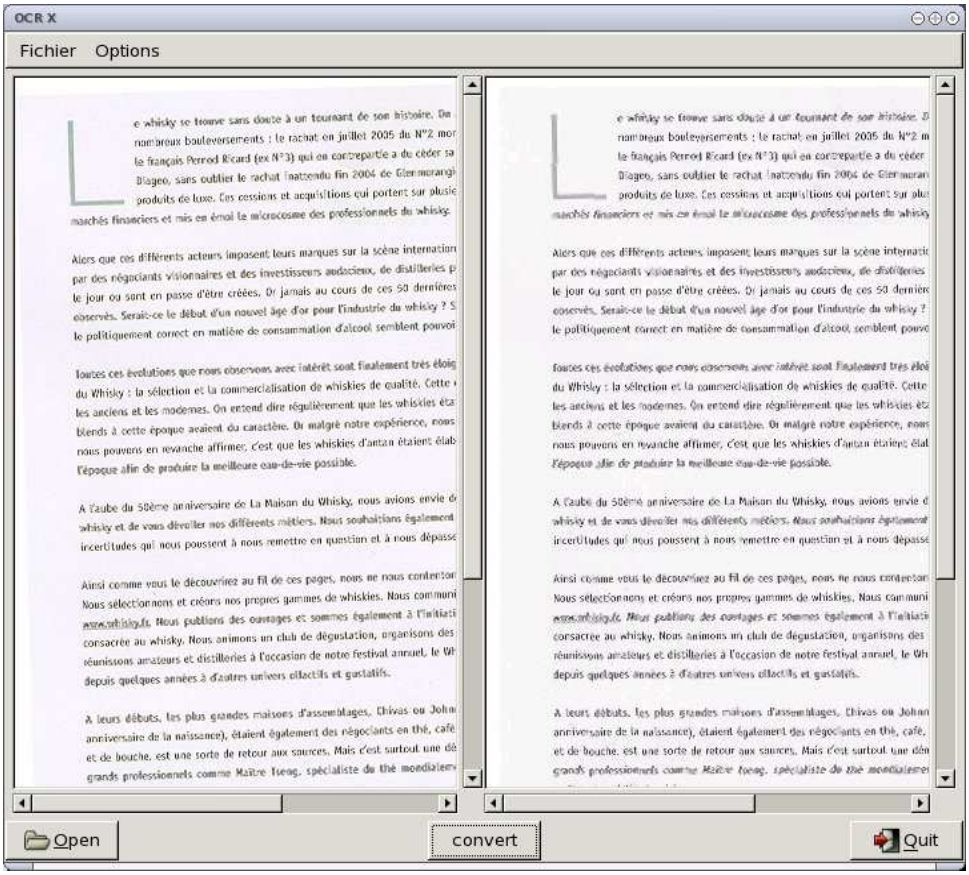
Mais le site web nous permettra bientôt aussi de nous tenir au courant dans une partie privée des modifications que nous apporterons au projet si nous travaillons séparément.

4. Projet

4.1. Interface

L'interface permet pour le moment de choisir l'image que l'on veut analyser et de la visualiser dans la partie gauche de l'interface. Une fois l'image choisie, on peut lancer le traitement et visualiser dans la partie droite le résultat.

Une fois la reconnaissance des caractères effectuée, la partie droite permettra de modifier le texte reconnu pour corriger les erreurs éventuelles.

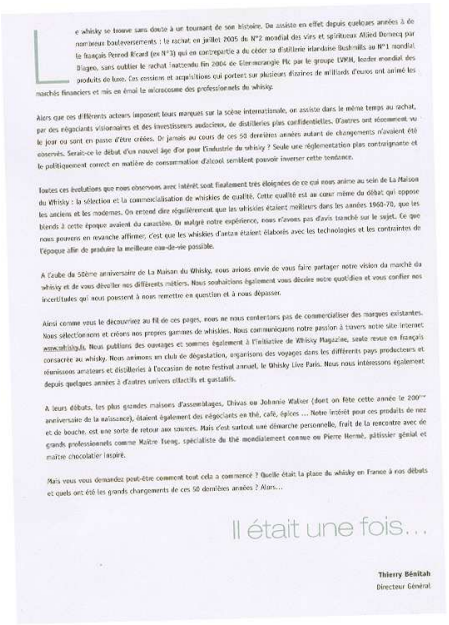


4.2. Réduction du bruit et rotation

La réduction du bruit permet d'obtenir une image ou le texte ressort mieux et est donc plus facilement analysable. Une fois le bruit retiré on redresse l'image si elle n'était pas droite. On peut difficilement voir la différence, sur les captures d'écran, une fois le bruit retiré, sans zoomer au maximum sur l'image mais on voit le résultat de la rotation.

Les algorithmes sont décrits dans la chronologie individuelle.

Original :



Il était une fois...

Thierry Bénitah
Directeur Général

parfaite :

Il était une fois...

Thierry Bénitah
Directeur Général

5. Chronologie

5.1. Chronologie (Groupe)

Nous avons tous commencé par faire des recherches sur le principe d'un OCR et les différentes méthodes de traitement d'image pour rendre l'image « parfaite », c'est-à-dire éliminer les bruits et les taches.

Puis nous avons fait des recherches sur les librairies permettant de loader et de manipuler des images. Nous avons trouvé Imlib et Imagemagick. Notre choix s'est penché sur Imlib2 car la librairie était mieux documentée sur Internet.

Une fois décidé sur le choix de la librairie nous avons mis en relation nos différentes idées sur le traitement de l'image : réduction du bruit, passage en noir et blanc, détection de l'angle et rotation de l'image.
Nous avons retenu des algorithmes intéressants.

Une fois ces étapes faites, nous nous sommes séparés les taches et avons décidé de travailler à l'aide du SVN de l'école.

Xavier devait s'occuper de la correction du bruit.

Kevin de l'interface graphique et de la bibliothèque gtk.

Jocelyn de la mise en mémoire de l'image ainsi que des fonctions de manipulations du vecteur de pixel fournit par imlib

Loïc de la rotation de l'image ainsi que de l'aide dans les différentes étapes ayant déjà eu l'occasion faire du traitement d'image.

Les 4 membres se sont aussi penchés sur les algorithmes de réduction de bruit ensemble.

5.2. Chronologie (individuelle)

5.2.1. Xavier

Xavier s'est occupé de la réduction du bruit sur l'image. Les algorithmes ont été trouvés et débuggés en groupe !

L'algorithme que nous avons tout d'abord implémenté consistait à prendre un seuil fixé arbitrairement et de dire que tous les pixels étant supérieur à ce seuil deviendraient blancs et que tous les autres deviendrait noirs. Cela nous a permit de nous rendre compte plus en détail du problème de bruit. En effet, sur l'image d'origine, il n'y a aucun pixel parfaitement égal à la couleur de son voisin, il y a énormément de nuances d'un pixel à l'autre. Cet algorithme n'a pas été retenu car ce procédé nous ferait perdre beaucoup trop en qualité, et lors de la détection de caractère, le travail serait plus difficile avec une image moins bonne.

Apres cette essai nous avons décidé d'étudier les pixels non plus 1 à 1 mais par matrice, c'est-à-dire un pixel plus les huit qui l'entourent. Le premier essai de ce nouvel algorithme consistait à énumérer les différentes couleurs autour du pixel étudié et de lui mettre la couleur la plus représentée avec un certain seuil.
Cet algorithme un peu plus performant que le précédent n'était quand même pas suffisant. Il restait en effet encore beaucoup de bruit, le fond n'était toujours pas assez blanc et les lettres pas assez apparentes.

L'algorithme retenu est basé sur le même principe que le deuxième, c'est-à-dire une matrice de pixels. Si les pixels autour de celui que l'on analyse sont plus ou moins identiques, au niveau du code R G B, en fonction d'un seuil fixé (+ ou – 50), ils sont notés comme bon. Une fois tous les pixels autours vérifiés, on applique au pixel analysé (au centre de la matrice) la moyenne des couleurs R G B des pixels retenus. Cet algorithme est beaucoup plus performant que le 2eme et corrige beaucoup de problème de bruit. L'image ressort nettement plus propre, sans perte d'informations utiles !!

5.2.2. Kévin

Kevin s'est tout d'abord renseigné sur la bibliothèque GTK et son utilisation pour la construction de l'interface. Il a cherché à comprendre le fonctionnement de ce que l'on appel des callback : association d'une fonction à un bouton.

Une fois cette étape faite il c'est attaqué à la construction de l'interface en elle-même. Pour cela il a fallu comprendre le principe des « conteneurs » que l'on ajoute à la fenêtre principale dans lesquels on met ensuite les boutons etc.

Une fois l'interface faite, Kevin c'est occupé de la détection de l'angle de rotation. L'algorithme consiste à initialiser un tableau de pixel à blanc ayant pour taille la hauteur de l'image. Il parcours ensuite chaque ligne de l'image et des qu'il rencontre un pixel noir, il met à noir la case correspondant au numéro de la ligne. Il fait cette étape pour tout les angles allant de -15° à 15° avec un pas de 0.5°. il sauvegarde à chaque fois l'angle pour lequel il y a le moins de pixel noir

5.2.3. Jocelyn

Jocelyn s'est occupé de la manipulation de la librairie Imlib2 pour le chargement de l'image en mémoire et l'utilisation de la structure nécessaire pour le projet (Bien qu'elle fut modifiée pour les besoins de chacun). Il s'est occupé de trouver comment utiliser les données récupérées grâce à la librairie sur l'image.

De plus, les fonctions de conversions de couleurs à partir de l'hexa, en Int, (et vice versa) ont aussi été codé par Jocelyn. Ces fonctions sont nécessaires pour assurer la compatibilité avec les formats de imlib.

Le passage de l'image en niveau de gris a aussi été codé par Jocelyn. Elle prend effet une fois que la réduction de bruit est achevée.

5.2.4. Loïc

Loïc s'est occupé de la rotation de l'image. Elle est réalisée grâce à une matrice de la forme :

$$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

Pour le moment la rotation n'est pas encore parfaite, en effet elle a tendance à déformer les mots par moment, mais Loïc est en train de réaliser une fonction d'interpolation de pixel pour améliorer le résultat après la rotation.

Loïc a aussi beaucoup aidé pour les algorithmes pour la réduction du bruit et la séparation des couleurs R G B ainsi que le débogage.

6. Amélioration à venir

6.1. Interface

Nous comptons ajouter une barre de loading pendant le traitement de l'image pour permettre de se rendre compte de l'avancement de la reconnaissance.

Nous allons aussi améliorer l'aperçu de l'image mise en mémoire pour une meilleure vue générale.

6.2. Réduction du bruit

Nous pensons améliorer le temps de traitement en utilisant des threads. Le traitement étant très efficace, il faudra peut être voir si l'on peut se permettre de réduire la définition de l'image quand elle excède une certaine taille !

6.3. Détection de l'angle de rotation

Nous pensons utiliser des threads pour factoriser le temps de détection de l'angle qui est pour l'instant trop important. Nous allons nous pencher sur la question, car notre programme prends le plus de temps à consacrer cette étape, c'est pour cela qu'il sera peut être nécessaire de coder un autre algorithme plus performant.

6.4. Rotation

Nous allons modifier l'algorithme en utilisant une interpolation de pixel car pour le moment le résultat n'est pas satisfaisant. En effet la qualité qui est résulte est amoindrie et, de ce fait, la détection des caractères ne pourra être que plus laborieuse !